Reinforcement Leaning Using a Gauss-Sigmoid Neural Network

Shin’ichi Maehara, Masanori Sugisaka, and Katsunari Shibata
Department of Electrical and Electronic Engineering, Oita University
700Dannoharu, Oita 870-1192, Japan. Email: maehara@cc.oita-u.ac.jp

Abstract

Boyan et al. has pointed out that the combination of reinforcement learning and Sigmoid-based neural network sometimes leads to instability of the learning. In this paper, it is proposed that a Gauss-Sigmoid neural network, in which continuous input signals are put into a Sigmoid-based neural network through a RBF network, is utilized for reinforcement learning. It is confirmed using simulation of the same task as in Boyan et al.[1] that the learning is faster and more stable when the Gauss-Sigmoid neural network is used, than when the Sigmoid-based neural network is used.

1 Introduction

Recently, the autonomous ability of reinforcement learning has attracted public's attention for the development of autonomous robots and learning machines. Reinforcement learning is generally used for action planning, and the machine learns the mapping from each state in the designed state space to an appropriate action.

By the combination of reinforcement learning and neural networks, a series of processes from sensors to motors including recognitions, can be synthetically learned[2]. Also, it is possible to acquire a continuous state space through learning in the hidden layers of the neural network, which can be utilized in another task.

However, Boyan et al. has pointed out that the combination of reinforcement learning and Sigmoid-based neural network sometimes leads to instability of the learning[1]. On the other hand, Gordon and Sutton showed that the instability of learning could be avoided by employing approximation methods based on the localization of input signals, such as CMAC, k-nearest-neighbor, and RBF (Radial Basis Function) [3][4][5]. If a task needs approximation of a strong non-linear function, then localizing continuous signals and using a representation like a table-look-up is effective. However, the output of such functions is represented as a linear sum of the localized signal, and they don't have hidden layers to represent global information, by integrating the localized signals adaptively. For example, when robots learn more than one task, the knowledge, which could be obtained from the previous sets of learning, is not utilized in present learning. Gaussian soft-max network[6] also utilizes RBF (Gaussian) units, and its generalization ability is an improvement on the regular RBF network. However, performance is not improved in areas where the RBF units are densely assigned.

In this paper, we use the neural network called the Gauss-Sigmoid neural network[7] and verify the stability in reinforcement learning for the hill-car task that Boyan et al. employed[1].

2 Hill-car problem

In this section, the hill-car problem is introduced as a problem in which the approximation of strong non-linear function is required. In this task, a car is located somewhere on the slope as shown in Fig.1, and must go up the right slope.

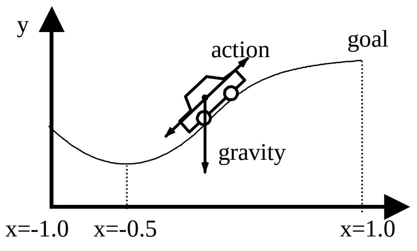


Fig.1 hill car problem.

The equation of the slope is described as

$$\begin{cases} y = x(x+1) & if \quad x < 0 \\ y = x/\sqrt{1+5x^2} & if \quad x \geq 0. \end{cases} \tag{1}$$

The equation of the car’s motion can be described as

$$\begin{cases} \dfrac{dv}{dt} = \{\dfrac{action}{m} - \dfrac{y'}{\sqrt{1+y'^2}}g\}/\sqrt{1+y'^2}, \\ \dfrac{dx}{dt} = v, \quad \dfrac{dy}{dx} = y', \end{cases} \tag{2}$$

where *x, y*: position of the car, *m*: mass of the car, *action*: driving force of the car *g*: gravity. In this problem, since the maximum driving force of the car is not so strong, the acceleration in the climbing direction around the steepest area of the slope is always negative, Therefore, when the car starts from (*x, v*)=(-0.5,0.0), the car cannot climb the right slope at one go even if it tries to climb with the maximum driving force. When the car is located on the right side of the boundary whether the car can climb the right slope at one go or not, the direction of the driving force should be right. When it is located on the left side of the boundary, the direction of driving force should be left because the car has to go up the left slope and then returns to the right slope with higher velocity. Thus, at this boundary, both the ideal value function and action function become discontinuous, and the approximation of these functions needs strong non-linearity.

3 Gauss-Sigmoid neural network

Sigmoid-based neural network (NN) have global generalization ability, since the output function of each unit is sigmoid function, but it is not suited for approximation of strong non-linear functions such as a step function. The output of RBF network is represented as a linear sum of the localized signal, however they don't have hidden layers to represent the global information by integrating the localized signals adaptively. Therefore, in this paper, we employ the Gauss-Sigmoid NN in which the output of RBF is used as the input of the hidden layer of sigmoid NN as shown in Fig.2.

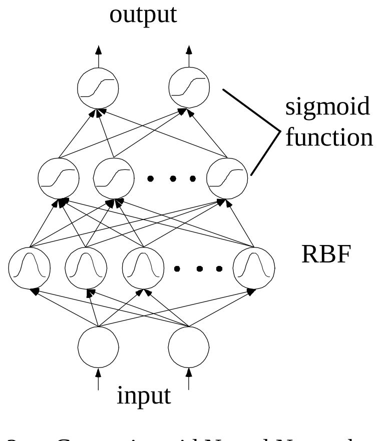


Fig.2 Gauss-sigmoid Neural Network.

Sigmoid NN itself is weak in the approximation of strong non-linear functions, but it becomes easy to approximate a strong non-linearity by using localized signals as inputs. Moreover, it is possible to obtain the global representation, through learning, by integrating the localized signals.

The output of RBF network is written as

$$output = \sum_{d=1}^{D} w_i g_i(x) + \theta, \quad \text{where} \tag{3}$$

$$g_i(x) = \exp(-\frac{1}{2}\sum_{d}^{D}(\frac{x_d - \mu}{\sigma})^2) \tag{4}$$

where $(\mu_{i,1},...\mu_{i,D})$: the center of the i-th RBF unit, $(\sigma_{i,1},...\sigma_{i,D})$: the size of the i-th RBF unit, θ : bias, *n*: the number of RBF units, *D* : the number of the input patterns. Both parameters of each RBF unit are also trained by the back-propagation learning method (BP), in the same way as the weights. However, when the size σ , is too small, the shape becomes steep and the updated value becomes too large. Then σ is converted to a logarithmic scale, and the update value is multiplied by the size σ . The update equation of the center μ is written as

$$\Delta\mu_{j,d} = g_j(x)\frac{x_d - \mu_{j,d}}{\sigma_{j,d}}\delta_j, \tag{5}$$

where δ_j : propagation error. In the training of size σ , *s* is defined as

$$\sigma_{j,d} = \exp(s_{j,d}), \tag{6}$$

and is updated as

$$\Delta s_{j,d} = g_j(x)\frac{(x_d - \mu_{j,d})^2}{\sigma_{j,d}^2}\delta_j. \tag{7}$$

Then *s* is transformed into the size σ by Eq.(6).

4. Actor-critic architecture

Q-learning and Actor-critic architecture are known as popular reinforcement learning algorithms. Here, Actor-critic architecture is employed as the paper of Boyan et al.. Actor-critic architecture is composed of an actor (action generation part) and a critic (state evaluation part). The critic evaluates the present state based on the past experiences of the system, and actor learns the action signal. In the critic, the previous state value is updated by using the present value to decrease TD (Temporal difference) error

$$\hat{r} = r_t + \gamma\ P(x_t) - P(x_{t-1}), \tag{8}$$

where γ : a discount factor, *r*: reward, x_t : input, $P(x_t)$: value. The update equation of the state value is written as

$$\Delta P(x_{t-1}) = \alpha_p \hat{r}_t, \tag{9}$$

where α_p : a learning rate of the value.

On the other hand, in the actor, $a(t)$ is the output and the actual action signal $\tilde{a}(t)$ is chosen from a stochastic distribution whose center is $a(t)$. Then, the action is updated to obtain more gain of the state value.

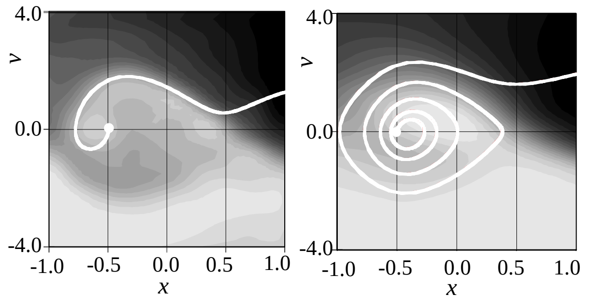
$$\Delta a(x_{t-1}) = \alpha_a(\tilde{a}_{t-1} - a(x_{t-1}))\hat{r}_t, \tag{10}$$

where α_a : a learning rate of the action. In this simulation, only one Gauss-Sigmoid NN is used for both the actor and critic. The network has two output units, one is for the action, and the other is for the value. If the action is not a scalar, the number of action outputs equals the number of elements of the action vector.

5. Simulation

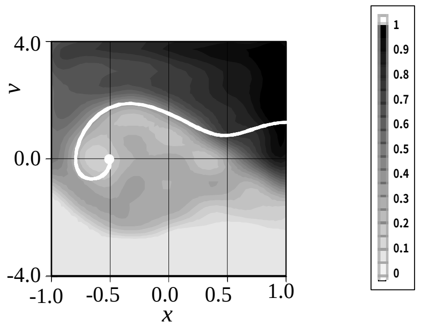
The hill-car problem is solved by actor-critic type reinforcement learning. Learning ability is compared in three cases; (1) Sigmoid-based NN, (2) Gauss-Sigmoid NN, and (3) RBF network. The initial state of the car is chosen from random numbers within the limits of $-1<x<1$ and $-4<v<4$. The actual driving force is the sum of the action, and the small uniform random number powered by 3. The state transition is calculated by solving the Eq. (2) by the Runge-Kutta method. When a car arrives at the top of the hill, the reward r =1.0, otherwise r =0.0. In the critic, the state value is updated by Eq. (9), but when the car rushes out to the left side of the slope or arrives at the top of the hill, $p(x_t)$ in Eq. (8) is 0. The velocity of the car was fixed at -4.0 when it became smaller than -4.0, and 4.0 when it became larger than 4.0. Since the driving force, *action*, was limited from -3.0 to 3.0, the car must go through the state of $x \leq -0.74$ with v=0.0 on the left side to arrive at the top. The value range of the sigmoid function is from -0.5 to 0.5. The Gauss-Sigmoid NN and Sigmoid-based NN used both a single hidden layer, and 40 neurons. The number of Gaussian units in the RBF network and Gauss-Sigmoid NN is 110(11×10). The learning of center and size of the Gaussian units aren't performed in this simulation. The learning rate for each network is 100/sqrt (the input number of each unit) in Gauss-Sigmoid NN, 80/sqrt (the input number of each unit) in Sigmoid-based NN, 0.48 in RBF network, and a momentum term was not used. The learning was iterated for 300,000 steps.

The value functions and the loci of the car in the hill-car task are shown as Fig.3. The initial state of the locus is (x, v)=(-0.5,0.0) which is the bottom of the slope. In all cases, the value is large in the upper right part where x and v are large. The value surface has a cliff around the boundary, whether a car can climb right slope at one go or not. The ridge of value function can be observed clearly around (x, v)=(-0.85,0.0), and the car can climb top of the hill after it goes up the left slope in the case of Gauss-Sigmoid NN and RBF network. While in the case of Sigmoid-based NN, the ridge of value function cannot be observed clearly, and the car cannot climb the slope unless it goes back and forth many times



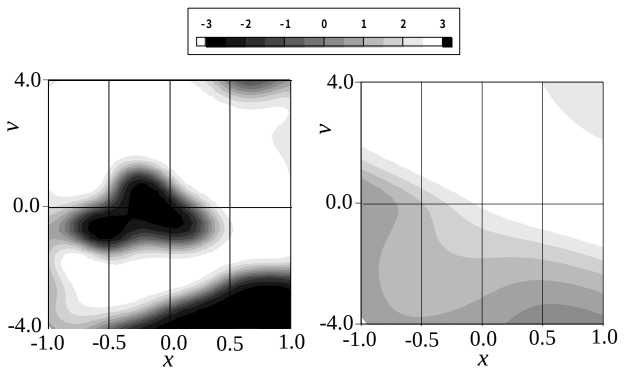
a) Gauss-Sigmoid NN (11×10 Gaussian) (40 sigmoid units)

b) Sigmoid-based NN (40 neurons)



c) RBF network (11×10 Gaussian)

Fig.3 Value function and locus of the car in hill-car task. The initial state of the locus is (x, v)=(-0.5, 0.0) that is the bottom of the slope.



a) Gauss-Sigmoid NN (11×10 Gaussian) (40 sigmoid units)

b) Sigmoid-based NN (40 neurons)

Fig.4 The magnitude and direction of the driving force.

Fig.4 shows that driving force as a function of the car's states. In the case of Gauss-Sigmoid NN, the boundary where the direction of the driving force changes can be clearly. It can be known that the direction of the driving force is left when the car is the lower part of the left slope with a small negative velocity and when the car is on the lower part of the right slope with a small velocity. While, in the case of Sigmoid-based NN, there are no clear regions where the direction of the driving force is left.

The learning curve in hill-car task is shown in Fig.5. The average number of steps to the goal over 37 initial states is plotted every 2000 steps. The 37 initial states are located on the grid at intervals of ($\Delta x, \Delta v$)=(0.25,

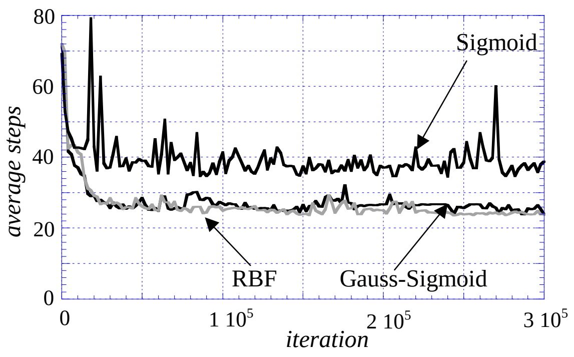


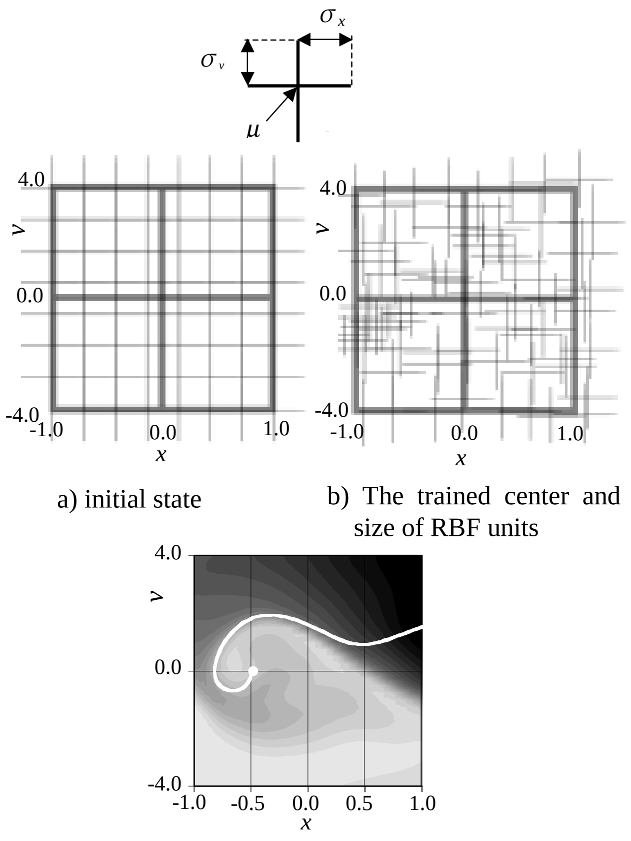
Fig.5 Comparison of the learning curve in the hill-car task.

1.33). The other 11 states, from which the car cannot climb the slope physically, are excluded. In Fig.4 it is shown that the learning speed and error of Gauss-Sigmoid NN is almost equal to RBF network, even though the sigmoid function is utilized. The error did not decrease as well in the Sigmoid NN. In the Sigmoid NN, the error did not change when the number of the hidden neurons was increased to 100.

Next, the center μ and size σ of each RBF unit is trained in the Gauss–Sigmoid NN. By this process, it is expected that efficient learning is performed and the same learning performance can be realized with fewer RBF units. Fig.6 shows the center and size of each of RBF unit, after learning, in the case of the Gauss-Sigmoid NN. The number of RBF units in this Gauss-Sigmoid NN is 64(8×8). The range of input signals is shown by a gray-framed rectangle in each figure. It is clear that many RBF units move to the place where strong non-linearity is required, and their sizes become small. When the parameters of RBF units were not trained, it could not reach the top at one go in all the 3 simulations in which the initial weights are varied, while it could reach in all the 3 simulations when the parameters were trained.

6. Conclusion

The paper proposed use of the Gauss-Sigmoid neural network, when the input signals represent continuous and global spatial information. In the hill-car task, it was shown that the performance was almost the same as RBF network and better than the as that of an performance of a Sigmoid-based neural network. Furthermore it was also shown that by learning the parameters of RBF units, the task could be solved with fewer RBF units.



a) initial state

b) The trained center and size of RBF units

c) Value function and locus of the car in case of moved RBF unit (8×8 Gaussian)

Fig.6 The change of center and size of RBF units, and Value function and locus of the car when the parameters of each RBF unit was trained in the Gauss-Sigmoid NN.

Reference

[1] Boyan, J.A. & Moore, A.W. : Generalization in Reinforcement Learning : Safely Approximating the Value Function, *Advances in Neural Information Processing Systems*, MIT Press, 7, pp.369-376(1995)

[2] Shibata, K. , Ito, K. & Okabe, Y. : Direct-Vision-Based Reinforcement Learning in "Going to an Target" Task with an Obstacle and with a Variety of Target Sizes, *Proc. of Inter. Conf. on Neural Networks and Their Applications '98*, PP.95-102(1998)

[3] Gordon, G. J. : Stable Function Approximation in Dynamic Programming, *Proc. of the 12-th ICML*, pp.261-268 (1995)

[4] Sutton, R. S. : Generalization in Reinforcement Learning: Successful Examples Using Space Coarse Coding, *In Advanced in Neural Information Processing System*, vol8, pp.1038-1044 (1996)

[5] Sutton, R.S. and Barto, A.G. : Reinforcement Learning, The MIT Press(1998)

[6] Morimoto, J. , Doya, K. : Learning Dynamic Motor Sequence in High-Dimensional State Space by Reinforcement Learning –Learning to Stand up-, *Proc. IEICE*, J82-D- Ⅱ ,No.11, pp. 2118-2131(1999) (in Japanese)

[7] Shibata, K. and Ito, K. : Gauss-Sigmoid Neural network, *Proc. of IJCNN'99*, #747(1999)